1983-03

Microcomputer laboratory design.

Abler, Ronald J.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MICROCOMPUTER LABORATORY DESIGN

by

Ronald J. Abler

March 1983

Thesis Advisor: Marle D. Hewitt

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Microcomputer Laboratory Design		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March, 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ronald J. Abler		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March, 1983
		13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer, Hardware, Software, Machine-language, Programming, Assembly-language, Programming.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A microcomputer laboratory was designed and implemented to support Airborne Digital Computation, AE 4641, a course involving a study of the methods used for digital computation in airborne weapons systems. Programming projects in both machine language and assembly language were written, tested, and provided for student exercise. All required operating and driving software was procured or written. Supporting notes, help files, and tutorials were composed.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Upon completion of the hardware and software implementation of the laboratory, the laboratory portion of the course was taught by the author, permitting rigorous testing and debugging of all equipment, operating systems, and programs.

Approved for public release, distribution unlimited

Microcomputer Laboratory Design

by

Ronald J. Abler
Commander, United States Navy
B.S., Loyola University, 1966
M.S., Prairie View A & M University, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1983

ABSTRACT

A microcomputer laboratory was designed and implemented to support Airborne Digital Computation, AE 4641, a course involving a study of the methods used for digital computation in airborne weapons systems. Programming projects in both machine language and assembly language were written, tested, and provided for student exercise. All required operating and driving software was procured or written. Supporting notes, help files, and tutorials were composed. Upon completion of the hardware and software implementation of the laboratory, the laboratory portion of the course was taught by the author, permitting rigorous testing and debugging of all equipment, operating systems, and programs.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

Airborne Digital Computation, AE 4641, was introduced into the Aeronautics program in 1974 as a study of the methods used for digital computation in airborne weapon systems. In fact, the course, as originally taught, concentrated exclusively on microprocessor technology. The laboratory consisted of a requirement for the students to complete a major programming project involving the design and implementation of an Intel 8080 microprocessor-based system to perform a particular (usually analog/digital) function. One Intel MDS-800 (Microcomputer Development System) computer, one Intel 8080 SBC (Single Board Computer), and associated hardware were purchased to support the course. Nevertheless, most projects became inordinately time-consuming since microprocessor-based system design tends to exceed by a wide margin the normal reasonable time frame consistent with the laboratory portion of a one-quarter graduate course. In addition, the students' entire laboratory experience was limited to a single specific and usually narrow application of the total technology.

For economic and personnel reasons, the course was discontinued in June, 1981. In June, 1982, the decision was made to reinstate the course and to offer it once again in the Fall Quarter of 1982. It was decided that a

radically different approach to the subject would be utilized, particularly in the laboratory portion of the course. In lieu of the single overwhelming assignment, a series of standardized projects of limited scope but of increasing complexity would be designed and implemented for assignment to the students. These projects were patterned after those assigned in the Computer Science Department's microcomputer course. The intent was to lead the students through a series of increasingly difficult projects, each building on the experience and accrued knowledge of its predecessor, resulting in a broad uniform exposure to microprocessor system development.

The objectives of this thesis, then, were:

1. To resurrect and restore to functionality the laboratory hardware.
2. To locate, install, revise, and update the software necessary to run the equipment.
3. To design, test, and implement the student projects.
4. To write the appropriate teaching and programming aids.
5. To develop a user-friendly operating system to minimize the impact of unfamiliarity with a new computer.

The material in this thesis is presented in basically the same order as that in which the design and implementation of the microlaboratory was accomplished, namely:

1. Laboratory Design

    a. Hardware implementation.

    b. Software installation.

    c. System upgrade and modification.

2. Course Development

    a. Student projects composition.

    b. Teaching and programming aids composition.

3. User-friendly Man-machine Interface

    a. Keyboard implementation of user keys.

    b. Menu-driven operating system composition and installation.

The students' experience with, and response to, the assigned projects, the learning aids, and the menu-driven operating system is discussed in the Results section. Recommendations for further improving the microlaboratory are included in the Conclusion. Appendix A contains the program listing for the demonstration program used to introduce Project #1. It is the only listing included in the thesis proper since it exists in otherwise unprintable form in PROM (Programmable Read-Only Memory) rather than on the class master disks because it is intended to run on the Prompt 80/85 classroom demonstration computer. Appendix B details the procedure for interfacing an aircraft control stick to the computer for use in Project #3 and in subsequent future thesis work. All programs necessary for the

conduct of the class, including the software written by the author, reside on the class master disks. A copy of the text files written to support the course has been assembled as a volume in the microlaboratory reference library. Solutions to the student projects have been provided for the AE 4641 instructor.

II. LABORATORY DESIGN

A. HARDWARE IMPLEMENTATION

The laboratory portion of the Airborne Digital Computation course, AE 4641, was implemented on existing, old but serviceable, equipment. Development started with the following equipment:

1. Intel MDS-800 (Microcomputer Development System) computer.
2. 16 kilobytes of random-access memory (RAM).
3. Datamedia Elite 2500 terminal.
4. Two single-sided, single-density Shugart 801 eight-inch floppy-disk drives.
5. Line 40 line printer.

B. SOFTWARE INSTALLATION

Locating, hooking up, and testing hardware resulted only in a shelf full of equipment which could do nothing useful. Software had to be written and installed to make the equipment functional.

Software installation included the following phases:

1. Establish a reference library.
2. Install the operating system.
3. Write and implement the printer/computer interface.

4. Install user programs for disk interface, file management, text editing, and programming.

Each of the steps outlined above is considered in turn below.

1. Reference Library

A complete survey of all existing software for the MDS-800 was conducted. Two file copies on disk of every available program and its documentation, if available, were made. Files and programs without documentation were retained for archival reasons, but no attempt was made to decipher their purpose or operation. A paper copy of every printable file was printed out. All hardware manuals were reviewed and filed. The floppy disks, paper copies, and manuals were sorted, collated, and then placed into the microlaboratory library. Table 1 lists the contents of the reference library as of this writing.

2. Operating System

The MDS-800 was initially implemented with the Aeronautical Engineeering Department's registered copy of the CP/M (version 1.3) operating system. Significant limitations in power and speed were immediately noted, and the decision was made to upgrade the operating system. This upgrade is discussed under the heading of System Upgrade and Modification.

Table 1. The Microlaboratory Reference Library.

1. Intellec 800 Microcomputer Development System Operator's Manual, #98-129A, Intel Corporation, 1975.

2. MDS-800 Intellec MDS Microcomputer Development System Hardware Reference Manual, #98-132B Rev B, Intel Corporation, 1975.

3. Using the 8251 Universal Synchronous/Asynchronous Receiver/Transmitter, AP-16, Intel Corporation, 1976.

4. Diskette Operating System -- MDS-DOS Operator's Manual, #98-206B, Intel Corporation, 1975.

5. MDS-DOS Hardware Reference Manual, #98-212A, Intel Corporation, 1975.

6. An Introduction to CP/M Features and Facilities, Digital Research, 1978.

7. CP/M Assembler (ASM) User's Guide, Digital Research, 1978.

8. CP/M Dynamic Debugging Tool (DDT) User's Guide, Digital Research, 1978.

9. ED: A Context Editor for the CP/M Disk System User's Manual, Digital Research, 1978.

10. CP/M Interface Guide, Digital Research, 1978.

11. CP/M System Alteration Guide, Digital Research, 1978.

3. Printer/Computer Interface

An existing public-domain program called PRINT.COM was dusted off and used to drive the Line 40 printer. It worked only to print ASCII files (restricted to those written under the TED word processor) from disk, but would not yet support screen dump or the control-P function from the keyboard.

4. User Programs

In order to standardize and simplify use of the computer system by students, class master disks were formatted and provided with all the programs that would be needed by the students for the class. The disks contain the previously discussed operating system, the complete portfolio of CP/M utility programs, a suite of public-domain programs, and numerous files written by the author to assist the students in operating the system and in completing the assigned programming projects.

a. CP/M Utilities

Table 2 lists the CP/M utilities provided on the class master disk. All of the utilities are provided by Digital Research as a standard part of CP/M 2.2. The file OVERVIEW.TXT provides an on-line thumbnail sketch of these and all other files on the disk.

b. Public-domain Programs

In order to expand the capability of the system, extend the power of the standard CP/M utilities, and provide a free, if rudimentary, text-editing facility, the public-domain programs listed in Table 3 were obtained from the CP/M User's Group in New Jersey and, in the case of the text editors TED and BTED, from the Computer Science Department.

Table 2. CP/M Utilities And Programs Provided On The Class Master Disks.

Program Name	Size	Description
ASM.COM	8K	8080 assembler.
CBIOS64.COM	2K	CP/M BIOS source file.
COPY.COM	8K	Disk-copying utility.
DDT.COM	5K	8080 debugger.
DISKREAD.COM	1K	Disk format-verifying utility.
DUMP.COM	1K	Memory-dump utility.
FORMAT.COM	1K	Disk-formatting utility.
LOAD.COM	2K	Hex-program-loading utility.
MAC.COM	12K	8080 Macro assembler.
PIP.COM	8K	File-transfer utility.
PRINT.COM		File-printing utility.
SID.COM	7K	8080 interactive debugger.
STAT.COM	6K	CP/M status utility.
SUBMIT.COM	2K	CP/M subroutine linker.
SYSGEN.COM	1K	CP/M operating-system generator.

Table 3. Public-domain Programs On The Class Master Disks.

Program Name	Size	Description
BTED.COM	13K	Faster version of TED.COM.
D.COM	2K	Improved CP/M directory utility.
FDIR.COM	8K	User-independent directory utility.
MANUAL.TED	11K	User's manual for TED and BTED.
PRO.COM	10K	Final processor for TED and BTED files.
RESTORE.COM	8K	Utility to restore erased files.
SORTDIR.COM	6K	Utility to alphabetize the directory.
SWEEP.COM	23K	Improved version of PIP.COM.
TED.COM	18K	Word processor (improved version of ED.COM).
TEDBUG.TXT	3K	List of known bugs in TED.

C. SYSTEM UPGRADE AND MODIFICATION

The first requirement was to increase the size of random-access memory by inserting three additional 16-kilobyte RAM cards into the MDS-800, for a total of 64 kilobytes of memory. This permitted the implementation of 62K CP/M (version 2.2) obtained from the Computer Science Department. Increased speed and a significantly more powerful command set were thus obtained.

Initial attempts to establish portability between the MDS-800 and other microcomputers (specifically the Altos computers located in both the Electrical Engineering and Computer Science Departments) were frustrated by incompatibility between disk drives. The Shugart 801 drives continually hung up when trying to read disks formatted and written to the IBM 3740 standard by either of the other two machines. The problem was solved by inserting a 100-picofarad silver-mica capacitor between pins 14 and 15 of chip number A4 on the floppy-disk-controller interface board, effectively increasing the bandwidth of the Shugart read circuit. The modified circuit diagram may be found on page 4-27 of the microlaboratory's copy of the MDS-DOS Hardware Reference Manual.

In order to provide the capability to print files other than those written under TED and BTED, PRINT.COM was rewritten to provide compatibility with other word processors. An original, unchanged copy has been retained for archival purposes, and a complete description of modifications is included in the source code of the revised version.

A new program, ECHOPRNT.COM, was written to implement keyboard control of, and screen dump to, the printer by use of the control-P command.

At this point, given 64K RAM, CP/M 2.2, inter-machine transportability, full printer capability, and a complete

suite of user programs, the microcomputer laboratory was ready for development of the Airborne Digital Computer course.

III. COURSE DEVELOPMENT

The first step in the specific development of the Airborne Digital Computation laboratory course was to set up the class master disks from which to teach the operation of the equipment; to demonstrate and implement the operation of CP/M; and to teach, demonstrate, and enable machine- and assembly-language programming.

In addition to the above, a goal of user-friendly operation was set so that, to the maximum degree possible, the course would be self-teaching and self-contained while minimizing the need for paper manuals, documentation, and notes. Course development thus consisted of the following sequence of tasks:

A. Write and install the three assigned student projects.

B. Write and install the help files which would explain and provide on-line assistance for computer operation, program use, and subroutine utilization.

C. Install SMARTKEY (copyright FBN Software) in order to facilitate software implementation of the user keys on the Datamedia Elite 2500 keyboard.

D. Install OKARA (copyright KIAĪ Systems) in order to permit programming of a user-friendly menu-driven CP/M.

E. Write and install the programs to implement menu-driven CP/M.

Each of these tasks is described in greater detail below.

A. STUDENT PROJECTS

1. Machine-language Multiplication

In Project #1 the student is required to write a machine-language program which will multiply two two-digit hex numbers and store the result in an assigned location and format. The purpose of the project is to introduce the student to the MDS-800 Development System and to assist the student in understanding both the Intel 8080 chip architecture and the 8080 assembly- language instruction set. To prepare the students for the project, a simple hex addition program was written and installed on the Prompt-80 classroom-demonstration computer. This program is demonstrated in the classroom as a part of the briefing for Project #1. The student's task is to expand the program into a workable, efficient multiplication scheme. Several files were written to support the demonstration and the project and are included on the class master disks. PROMPT80.TXT and PROMPT.LST provide instructions and a checklist, respectively, on the use of the machine to load and run the hex-addition demonstration program and to burn

it into an Intel 2708 PROM. Appendix I contains a listing of the demonstration program. PROJ1.TXT contains the project assignment and description. PROJ1.ASM and PROJ1.COM are the source and executable files, respectively, and have been given to the instructor.

2. *Simple I/O*

In Project #2, the student is required to write an assembly-language program which will accept ASCII input from the keyboard and print it to the CRT five times double-spaced. The purpose of this project is to introduce the concepts of, and develop proficiency in, basic input-output procedures. Again, several files were written to support the project and are included on the class master disks. PROJ2.TXT explains the assignment. PROJ2.ASM and PROJ2.COM are the source and executable files, respectively, and have been given to the instructor.

3. *Simple Graphics and Interrupts*

Project #3 offers three options. In all three options, the student must write an assembly-language program which will demonstrate facility with simple graphics and the use of interrupts in programming. PROJ3.TXT was written to explain the assignment and is included on the class master disk. The options are delineated below.

a. The first option requires the student to write a program which will initialize the counter-timer chip in the

MDS-800 and use it to regulate the display of a digital keyboard-controllable stopwatch on the CRT. PROJ3A.ASM and PROJ3A.COM are the source and executable files, respectively, and have been given to the instructor.

b. The second option requires the student to write a program which will move a single character around the CRT in a specified pattern as commanded from the keyboard. If an interrupt is generated while the character is within the confines of a defined target box, a "hit" is to be displayed on the CRT; otherwise, a "miss" is to be displayed. PROJ3B.ASM and PROJ3B.COM are the source and executable files, respectively, and have been given to the instructor. Appendix B details the specifications of an interrupt-triggering mechanism which has been wired into a military aircraft control stick. The control buttons embedded in the control stick may be used in lieu of the manual interrupt switches on the front panel of the MDS-800 to provide a more realistic "arcade" or cockpit environment.

c. The third option requires the student to write a program of his own design, subject to instructor approval, which will satisfactorily demonstrate the assigned objectives.

B. TEACHING AND PROGRAMMING AIDS

Table 4 lists the help files which were written. Included are both on-line assistance files and

supplementary files which contain the notes from lectures given to introduce laboratory operation and programming techniques.

Table 4. Programs And Files Written To Support Student Projects.

Program Name	Size	Description
ASM.TXT	4K	User's guide for the assembler.
CPM.TXT	8K	Introduction to CP/M.
CPMIO.TXT	2K	Introduction to CP/M input-output techniques.
CTC.TXT	5K	Instructions for initializing and programming the MDS-800 Counter-Timer Chip.
CURSOR.TXT	3K	Introduction to cursor control.
DDT.TXT	4K	User's guide for the debugger.
ECHOPRNT.COM	1K	Line 40 printer driver.
ECHOPRNT.MAC	3K	Source code for printer driver.
ECHOPRNT.TXT	2K	Instructions for use of ECHOPRNT.
GRAPHICS.TXT	5K	Introduction to ASCII graphics.
INTRO.TXT	4K	Course introduction.
OUTLINE.TXT	7K	Course outline.
OVERVIEW.TXT	6K	Program list and summary.
PRINT.COM	2K	File printing utility.
PROJ1.TXT	2K	Explanation of Project #1.

PROJ2.TXT	3K	Explanation of Project #2.
PROJ3.TXT	6K	Explanation of Project #3.
ROUTINES.TXT	2K	Program listing of usable subroutines.
SERIAL.TXT	4K	Introduction to serial I/O techniques.
TIMER.TXT	1K	Notes on how to program a timer.
VERSION.LST	3K	List of latest revision of all files and programs on the Class Master Disk.

IV. USER-FRIENDLY MAN-MACHINE INTERFACE

Two commercial software packages were purchased and installed on the MDS-800 to facilitate the development of the user-friendly man-machine interface. The first of these was SMARTKEY from FBN Software, which provides the capability of assigning to any discrete-coded key on the terminal the user's choice of characters and/or commands. The second program was OKARA from Kiai Systems, which functions as a CP/M "shell" and provides an interactive operating environment in which command menus and scripts may be implemented.

A. SMARTKEY

SMARTKEY was purchased and installed on the MDS-800. Complete documentation is contained in the accompanying manual [Ref. 1]. SHELL.DEF is the definition file which contains the user key assignments for the Datamedia Elite 2500 terminal. "SHELL" is invoked as the parameter in the SMARTKEY call to bind the user keys to the functions shown on the overlay labelled "Shell."

B. OKARA

OKARA was purchased and installed on the MDS-800. Complete documentation is contained in the accompanying manual

[Ref. 2]. OKARA permits warm-boot calls to be trapped rather than being automatically passed to CP/M. Then, depending on the source of the warm boot, action other than a warm boot, such as contained in menu-driven command scripts, can be executed.

C. MENU-DRIVEN CP/M

Given the unique features of OKARA, command scripts (designated by the filetype .KOM) can be written to provide true user-friendly menu-driven CP/M. Table 5 lists the .KOM files which constitute the menu suite, along with a brief explanation of their functions.

To facilitate the use of menu-driven CP/M for those students who were unfamiliar with CP/M while not slowing down those who already knew CP/M or who preferred to learn it without the help of a menu, a separate, totally independent class master disk, labelled "Okara Version" was provided. Table 6 lists the directories of the unmodified CP/M class master disk. The "Okara Version" class master disk contains all of these files plus the additional files listed in Table 7.

With the successful implementation of a truly user-friendly operating system, the goal of an essentially self-taught, self-contained microcomputer laboratory, free of the drudge of paper documentation and manual, was accomplished.

Table 5. The Menus Which Make Up The Menu-driven CP/M Facility.

Program Name	Size	Description
MENU$CPM.KOM	2K	Main menu from which all subordinate menus may be selected.
COPYFILE.KOM	2K	Menu from which file transfer functions can be executed.
COPYDISK.KOM	3K	Menu from which disks can be formatted, copied, and provided with the operating system.
STATS.KOM	7K	Menu from which disk, file, and drive status may be queried and/or altered.
TEXT.KOM	5K	Menu from which text files may be printed out or displayed on the CRT.
TREDS.KOM	5K	Menu from which files may be Typed, Renamed, or Erased, the Directory displayed, and blocks of memory Saved.

Table 6. Directory Of The Unmodified CP/M Class Master Disk.

Program Name	Size	Program Name	Size
ASM.COM	8K	OUTLINE.TXT	7K
ASM.TXT	4K	OVERVIEW.TXT	6K
BALL.TXT	1K	PIP.COM	8K
BTED.COM	13K	PRINT.COM	2K
COPY.COM	8K	PRO.COM	10K
CPM.TXT	8K	PROJ1.TXT	2K
CPMIO.TXT	2K	PROJ2.TXT	3K
CURSOR.TXT	3K	PROJ3.TXT	6K
D.COM	1K	RESTORE.COM	8K
DDT.COM	5K	ROUTINES.TXT	2K
DDT.TXT	4K	SERIAL.TXT	4K
ECHOPRNT.COM	1K	SORTDIR.COM	6K
ECHOPRNT.TXT	3K	STAT.COM	6K
FDIR.COM	8K	SUBMIT.COM	2K
FORMAT.COM	1K	SYSGEN.COM	1K
GRAPHICS.TXT	5K	TED.COM	18K
INTRO.TXT	4K	TEDBUG.TXT	3K
LOAD.COM	2K	TIMER.TXT	1K
MANUAL.TED	11K	VERSION.LST	3K

Table 7. Additional Files Contained On The "Okara Version" Class Master Disk.

Program Name	Size	Program Name	Size
COPYDISK.COM	3K	SET.COM	4K
COPYFILE.COM	2K	SHELL.DEF	1K
DO.COM	4K	STATS.KOM	7K
MENU$CPM.KOM	2K	TEXT.KOM	5K
OKARA.COM	4K	TREDS.KOM	5K
PAGE.COM	2K		

V. RESULTS

Student response to the design of the microlaboratory and its facilitating software was closely and personally monitored on an almost daily basis. A log was made available in which students were asked to note any bugs in the operating system or supporting programs, breakdowns of the hardware, errors of fact or grammar in the text files, inconsistencies in instructions and checklists, and recommendations or suggestions for improvement. The log was checked at the beginning of each school day. After completion of the course, the log contained two entries. Both were corrected on the same day of entry. One involved a bug in one of the input/output subroutines, and the other was the result of a crashed class master disk. The former was fixed by substituting a pair of input/output port addresses. The latter was rectified by reformatting and recopying the master disk. There were no hardware or operating software failures or deficiencies noted.

The response to SMARTKEY and to menu-driven CP/M (facilitated by OKARA) was mixed. The students used SMARTKEY consistently and with uniform approval as to its effectiveness and utility. Of the two keyboard overlays provided, the one implementing word processing functions was heavily utilized. The overlay designed for menu-driven

CP/M was less used for reasons discussed below. A significant increase in the use of SMARTKEY would probably be effected if a copy of the manual were placed in the reference library and the customizing subroutines put on the class master disks. Students would very likely develop their own personalized function-key assignments and thus extend SMARTKEY's utility in the course. It must be noted, however, that piracy of the software and violation of the department's copyright contract would be highly probable.

Menu-driven CP/M was a disappointment. Student use of the program was very low. There are no known bugs in the program. It does, however, run slowly, due mostly to the 2 MHz clock in the MDS-800. (Most modern microprocessors run between 4 and 8 MHz.) An even more enlightening explanation for low utilization of the program is that all of the students, for reasons of their own, preferred to learn CP/M. Therefore, it can only be concluded that menu-driven CP/M has not been given an adequate test from which to judge its merits. It is recommended that it continue to be provided for future classes. Its presence in no way hinders the operating system. It may yet meet the need for which it was written, given one or more students who have no requirement nor desire to struggle with CP/M for its own sake.

The three assigned student projects appear to have served their purpose. There are no known bugs remaining,

and all necessary subroutines have been provided. This conclusion is based on student estimates that the programming load did not exceed the course goal of ten hours per week, at least for the six students (out of eight) who turned in all projects in accordance with the assigned schedule. It is believed that the student projects did in fact provide the students with the desired hands-on familiarity with machine- and assembly-language programming, simple I/O procedures, basic graphics techniques, and the use of interrupts.

VI. CONCLUSION

Airborne Digital Computation, AE 4641, is a worthwhile course that fills a need for knowledge and experience that is not currently met by any other course provided by the Aeronautical Engineering Department. Insofar as it addresses specifically the applications of microprocessor technology in airborne weapon systems, it covers material not taught by any other department at the Naval Postgraduate School. It must be noted, however, that the hardware presently installed in the microlaboratory is outdated and outmoded. It cannot reasonably be expected to continue to function without breakdown indefinitely. There are no back-ups in the department, so a hardware failure can be catastrophic.

Terminal workload is a factor which must be considered. There were eight students using the single available terminal. Terminal utilization was high, but tolerable. However, it was apparent that a single terminal could not reasonably have handled any more students. As it was, the instructor found it nearly impossible to secure terminal time for himself during normal school hours.

It is recommended that the microlaboratory be improved and expanded, both in support of AE 4641 and of future thesis work which could be accomplished in the field of

airborne digital computation. It is not recommended that extensive thesis research be undertaken on the presently installed dinosaur.

Given the current state of the art in microprocessors, it is the opinion of this author that the following equipment should be obtained for the microlaboratory:

1. One S-100-bussed Z-80 microprocessor for every six class students or every two thesis students.
2. One 12-slot motherboard for each microprocessor.
3. One terminal for each microprocessor.
4. Two double-density, double-sided 8" floppy-disk drives for each terminal.
5. One hard-disk drive for each four terminals.
6. A fully networked operating system.
7. The following categories of software, in addition to the suite of utilities which should accompany the operating system:
    a. A word processing program.
    b. A relational data base program.
    c. A regular assembler and a macro assembler.
    d. A disassembler.

APPENDIX A

PROMPT 80/85 DEMONSTRATION PROGRAM

The following machine-language program, written in hex, adds two six-digit hex numbers and stores the result in registers 3CB0 through 3CB3 of the Prompt 80/85. The program is intended to be very simple and straightforward since its purpose is to demonstrate the architecture of the Intel 8080 chip and the manipulation of its registers. It is not intended to illustrate tight, efficient code.

This program has been burned into an Intel 2708 PROM (Programmable Read-Only Memory) whose starting address when inserted into the Prompt 80/85 is 3C00. Once the PROM has been inserted, the program is ready to demonstrate in accordance with the checklist delineated in PROMPT80.LST.

Address	Hex	Mnemonic	Comment
3C00	01	LXI B, 0193	Load LSByte of addres-
3C01	93		ses of numbers to be
3C02	01		to be added in register
3C03	11	LXI D, A3B3	B, C, D, E.
3C04	B3		
3C05	A3		
3C06	21	LXI H, 3C00	Load page number.
3C07	00		
3C08	3C		
3C09	69	MOV L, C	
3C0A	7E	MOV A, M	EF in accumulator.
3C0B	6A	MOV L, D	
3C0C	86	ADD M	EF + 65 = 154
3C0D	6B	MOV L, E	
3C0E	77	MOV M, A	54 stored in 3CB3
3C0F	0D	DCR C	Decrement C, D, E.

```
3C10    15    DCR D
3C11    1D    DCR E
3C12    69    MOV L, C
3C13    7E    MOV A, M      CD in accumulator.
3C14    6A    MOV L, D
3C15    8E    ADC M         CD + 87 + CY = 155
3C16    6B    MOV L, E
3C17    77    MOV M, A      55 stored in 3CB2.
3C18    0D    DCR C         Decrement C, D, E.
3C19    15    DCR D
3C1A    1D    DCR E
3C1B    69    MOV L, C
3C1C    7E    MOV A, M      AB in accumulator.
3C1D    6A    MOV L, D
3C1E    8E    ADC M         AB + B9 + CY = 165
3C1F    6B    MOV L, E
3C20    77    MOV M, A      65 stored in 3CB1.
3C21    1D    DCR E         Point to 3CB0.
3C22    DA    JC 3CC1       If carry, put 1 in
3C23    C1                  3CB0.
3C24    3C
3C25    FF    RST 7
.
.
.
3C91    AB
3C92    CD
3C93    EF
.
.
.
3CA1    B9
3CA2    87
3CA3    65
.
.
.
3CB0
3CB1
3CB2
3CB3
.
.
.
3CC1    6B    MOV L, E      If carry, put
3CC2    70    MOV M, B      1 in 3CB0.
3CC3    FF    RST 7
```

APPENDIX B

INTERRUPT-TRIGGERING MECHANISM

An aircraft control stick has been wired to the MDS-800 computer so that its three control buttons trigger interrupts in the computer. The top red pushbutton switch controls Interrupt #1; the trigger-switch, Interrupt #3; and the lower red pushbutton switch, Interrupt #5. The control stick is connected to the computer by two unique plugs which fit corresponding sockets on the back panel of the MDS-800 and which cannot be improperly connected. Once connected, the three control switches perform exactly the same functions as the corresponding rocker switches on the front panel of the MDS-800. The switches all operate independently, and all are functional when the control stick is connected.

LIST OF REFERENCES

1. SMARTKEY User's Manual, FBN Software, 1982.
2. OKARA User's Manual, Kiai Systems, 1982.

BIBLIOGRAPHY

8080/8085 Assembly Language Programming Manual, #9800301C, Intel Corporation, 1978.

Intel Prompt 80/85 User's Manual, #9800307C, Intel Corporation, 1978.

Intel 8080 Microcomputer Systems User's Manual, #98-153C, Intel Corporation, 1975.

Intel 8080 Assembly Language Programming Manual, #98-004C Rev. C, Intel Corporation, 1976.

SBS 80/10 Single Board Computer Hardware Reference Manual, #98-230B, Intel Corporation, 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 67Ln Department of Aeronautical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Marle D. Hewett, PhD, Code 67Hj Department of Aeronautical Engineering Naval Postgraduate School Monterey, California 93940	1
5.	Mr. Ted Dunton, Code 67Dtn Department of Aeronautical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	CDR Ronald J. Abler, USN 1022 11th. St. Manhattan Beach, California 90266	1

Thesis
A165 Abler 200627
c.1 Microcomputer labor-
atory design.